MÓDULO 5:
ENVIO DE VARIABLES ENTRE PAGINAS

Eduard Lara

INDICE

1. Envío de variables entre páginas PHP
2. Envío de variables desde formularios HTML
3. Acceso a elementos del formulario desde PHP
4. El formulario de PHP
5. Subida de ficheros al servidor
6. Validacion de formularios

1. ENVIO DE VARIABLES ENTRE PAGINAS

- El envío de variables entre páginas es una forma habitual de trabajar en scripts de servidor
  - procesado de formularios
  - carrito de compra).
- Se puede hacer por dos métodos:
  - GET
  - POST
- Acceder a los datos HTTP POST o GET es simple en PHP
- Las variables globales $_POST[] y $_GET[] contienen los datos de la petición HTTP

1. ENVIO DE VARIABLES ENTRE PAGINAS

Método GET

❖ Las variables y sus valores se pasan en la cabecera del mensaje HTTP junto con la URL de la página que debe aceptar dichas variables:

pagina.php?variable1=valor1&variable2=valor2

Página web pedida | Bloque variable = valor | Bloque variable = valor

❖ Los bloques variable=valor pueden ser tantos como se quieran, con la única restricción que impone el método GET de 2000 caracteres.

1. ENVIO DE VARIABLES ENTRE PAGINAS

Institut Joan d'Àustria

Método GET

- Los nombres de variables no van precedidos del signo $.
- Los valores de las variables no van escritos entre comillas (sean del tipo que sean).
- La página que recibe la información, puede acceder a las variables mediante

$_GET[variable1]

$_GET[variable2]

1. ENVIO DE VARIABLES ENTRE PAGINAS

Método POST

- Este método esconde las variables y sus valores en el cuerpo del mensaje HTTP (las variables no son visibles en la URL).
- El acceso a las variables se podrá hacer con:

$_POST[variable1]

1. ENVIO DE VARIABLES ENTRE PAGINAS

Variables globales

- A partir de PHP 4.2.0, el valor por defecto de la directiva de PHP register_globals es off
- Si la directiva register_globals del servidor se pone a ON, las variables podrán ser recogidas directamente mediante $variable1 (tanto para el método GET como para POST). como variables globales
- Se puede poner register_globals = on en el fichero de configuración php.ini, pero no es recomendable por motivos de seguridad.

1. ENVIO DE VARIABLES ENTRE PAGINAS

Método REQUEST

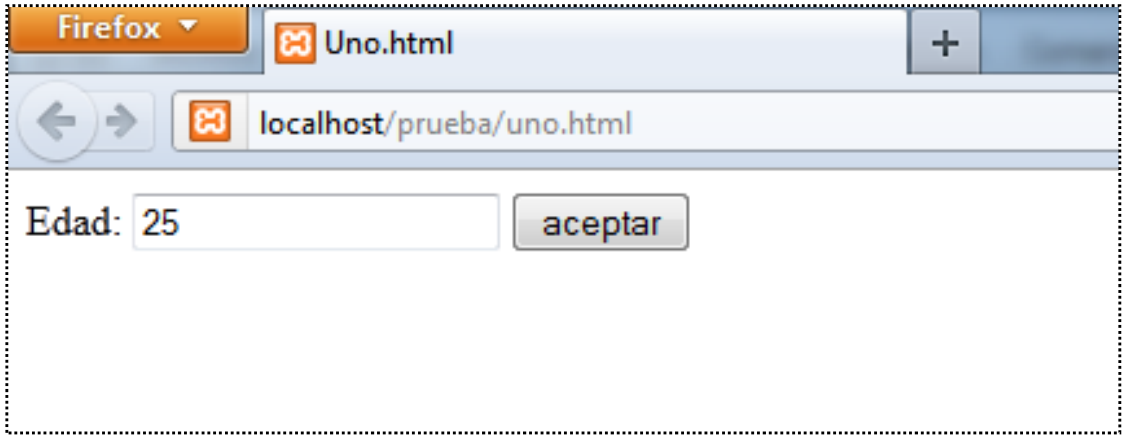
- Es un método comodín que captura tanto parámetros pasados por GET ó POST.
- La sintaxis es:

```
$edad = $_REQUEST['edad'];
```

2. ENVIO VARIABLES EN FORMULARIOS HTML

Fichero uno.html

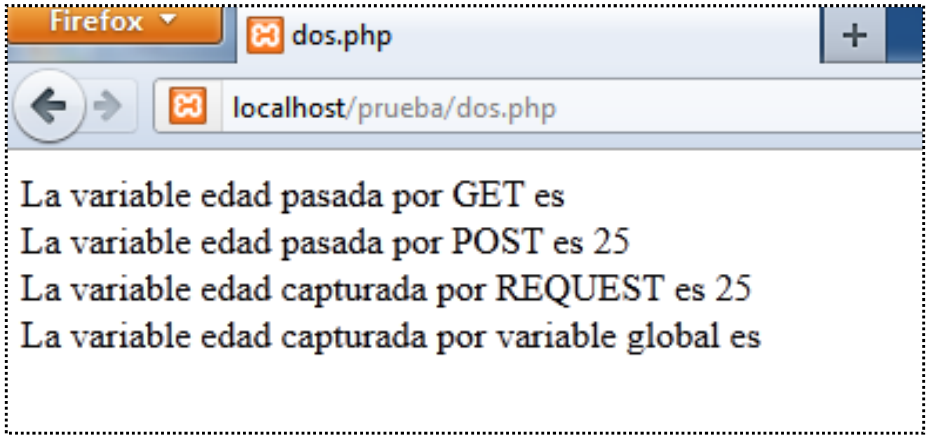
```
<HTML>
<BODY>
<FORM ACTION="dos.php" METHOD="POST">
    Edad: <INPUT TYPE="text" NAME="edad">
    <INPUT TYPE="submit" VALUE="aceptar">
</FORM>
</BODY>
</HTML>
```



2. ENVIO VARIABLES EN FORMULARIOS HTML

- Fichero dos.php

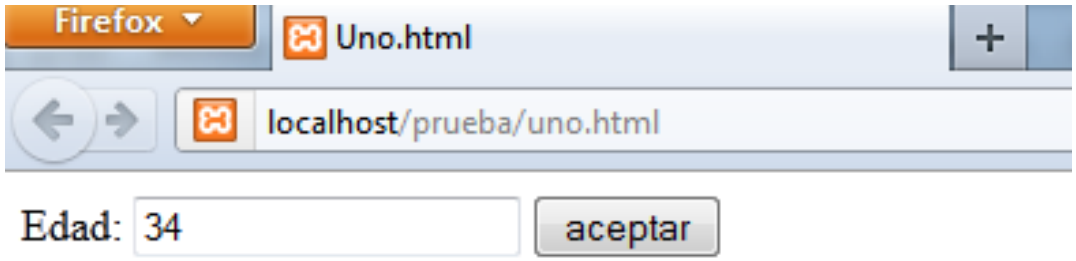
```
<HTML><BODY>
<?PHP
    echo "La variable edad pasada por GET es ".$_GET['edad']. "<BR>";
    echo "La variable edad pasada por POST es ".$_POST['edad']."<BR>";
    echo "La variable edad capturada por REQUEST es
    ".$_REQUEST['edad']."<BR>";
    echo "La variable edad capturada por variable global es ".$edad."<BR>“;
?>
</BODY>
</HTML>
```



2. ENVIO VARIABLES EN FORMULARIOS HTML

Fichero uno.html

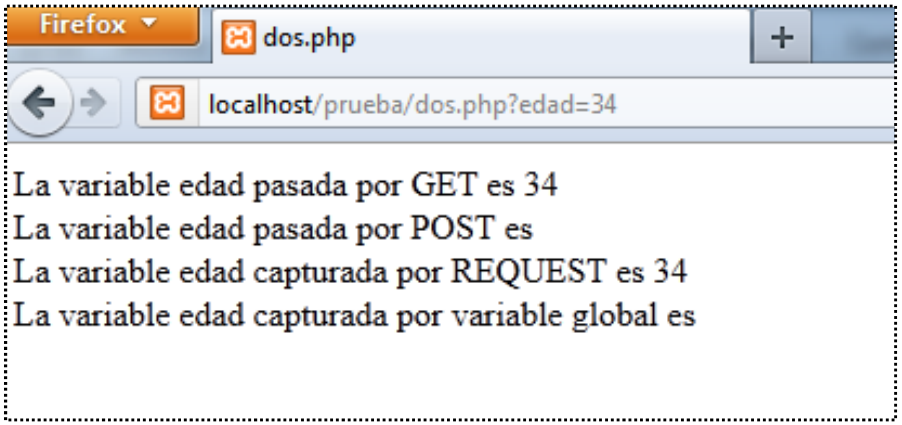
```
<HTML>
<BODY>
<FORM ACTION=”dos.php” METHOD=”GET”>
   Edad: <INPUT TYPE=”text” NAME=”edad”>
   <INPUT TYPE=”submit” VALUE=”aceptar”>
</FORM>
</BODY>
</HTML>
```



2. ENVIO VARIABLES EN FORMULARIOS HTML

- Fichero dos.php

```
<HTML><BODY>
<?PHP
  echo "La variable edad pasada por GET es
  ".$_GET['edad']. "<BR>";
  echo "La variable edad pasada por POST es
  ".$_POST['edad']."<BR>";
  echo "La variable edad capturada por REQUEST es
  ".$_REQUEST['edad']."<BR>";
  echo "La variable edad capturada por variable global es
  ".$edad."<BR>";
?>
</BODY>
</HTML>
```



3. ACCESO A ELEMENTOS FORMULARIOS

Acceso a los diferentes tipos de elementos de entrada de un formulario

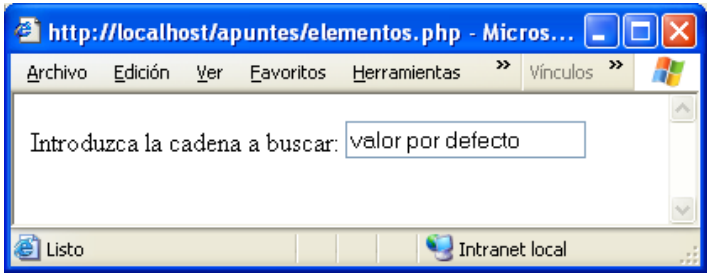
- Elementos de tipo INPUT
  - TEXT
  - RADIO
  - CHECKBOX
  - BUTTON
  - FILE
  - HIDDEN
  - PASSWORD
  - SUBMIT
- Elemento SELECT
  - Simple / múltiple
- Elemento TEXTAREA

3. ACCESO A ELEMENTOS FORMULARIOS

TEXT

Introduzca la cadena a buscar:
<INPUT TYPE="text" NAME="cadena" VALUE="valor por defecto"
SIZE="20">

```
<?PHP
   print ($cadena);
 //print ($_REQUEST ['cadena']);
?>
```



3. ACCESO A ELEMENTOS FORMULARIOS

RADIO

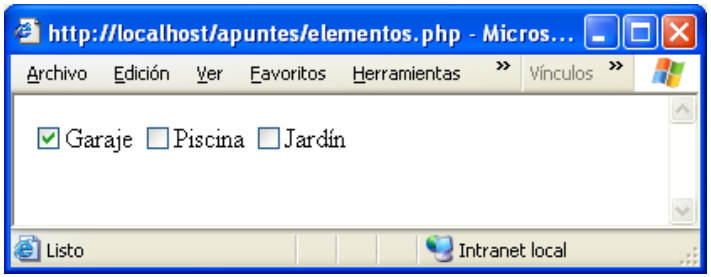
```
<INPUT TYPE="radio" NAME="titulacion" VALUE="II"
CHECKED>I.Informática
<INPUT TYPE="radio" NAME="titulacion" VALUE="ITIG">I.T.I. Gestión
<INPUT TYPE="radio" NAME="titulacion" VALUE="ITIS">I.T.I. Sistemas

<?PHP
   print ($titulacion);
 //print ($_REQUEST ['titulacion']);
?>
```

3. ACCESO A ELEMENTOS FORMULARIOS

CHECKBOX

```
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="garaje"
CHECKED>Garaje
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="piscina"> Piscina
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="jardin"> Jardín
```

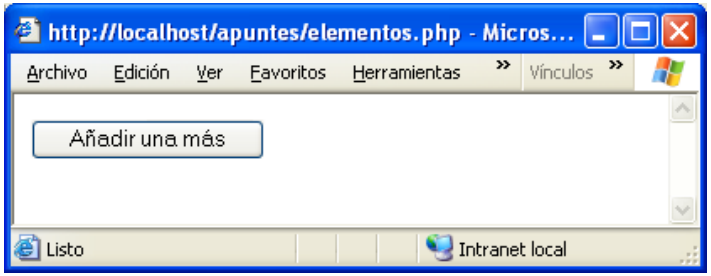
```
<?PHP
   $n = count ($extras);
   for ($i=0; $i<$n; $i++)
      print ("$extras[$i]<BR>\n");
 //foreach ($_REQUEST['extras'] as $extra)
    //print ("$extra<BR>\n");
?>
```



3. ACCESO A ELEMENTOS FORMULARIOS

BUTTON

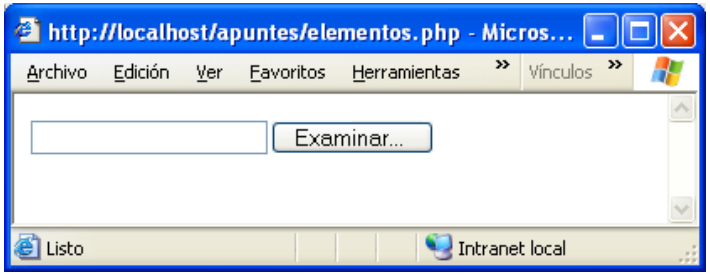
```
<INPUT TYPE="button" NAME="nueva" VALUE="Añadir una más">

<?PHP
   if ($nueva)
      print ("Se va a añadir una nueva");
   //if ($_REQUEST [‘nueva’])
      //print ("Se va a añadir una nueva");
?>
```



3. ACCESO A ELEMENTOS FORMULARIOS

FILE

```
<FORM ACTION="procesa.php" METHOD="post“
                                                ENCTYPE="multipart/form-data">
   <INPUT TYPE="file" NAME="fichero">
</FORM>
```

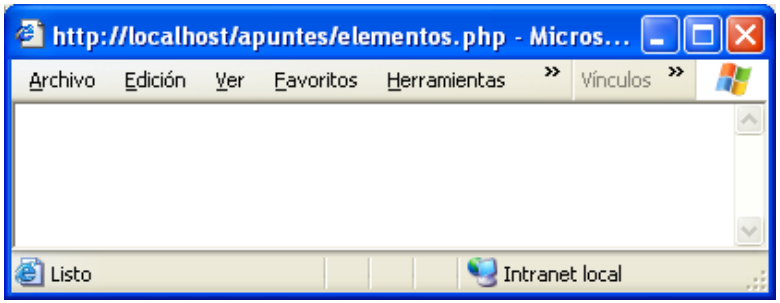


3. ACCESO A ELEMENTOS FORMULARIOS

HIDDEN

```
<?PHP
   echo “<INPUT TYPE=’hidden’ NAME=’username’ VALUE=’$usuario’>\n”;
?>
```

```
<?PHP
   print ($username);
   //print ($_REQUEST [‘username’]);
?>
```

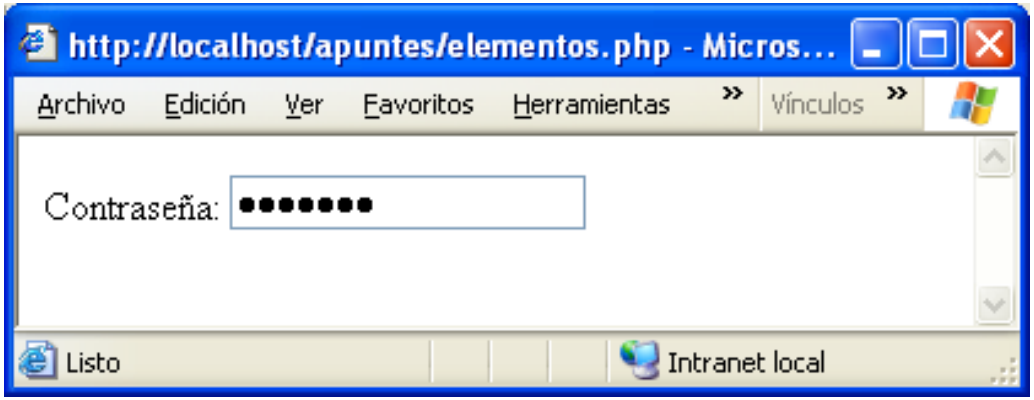


3. ACCESO A ELEMENTOS FORMULARIOS

PASSWORD

Contraseña: <INPUT TYPE="password" NAME="clave">

```
<?PHP
   print ($clave);
   //print ($_REQUEST [‘clave’]);
?>
```

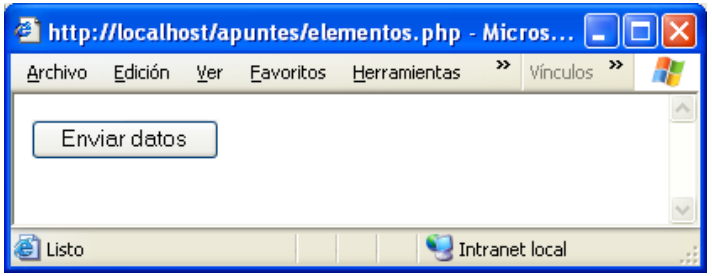


3. ACCESO A ELEMENTOS FORMULARIOS

SUBMIT

```
<INPUT TYPE="submit" NAME="enviar" VALUE="Enviar datos">
```

```
<?PHP
   if ($enviar)
      print ("Se ha pulsado el botón de enviar");
   //if ($_REQUEST ['enviar'])
      //print ("Se ha pulsado el botón de enviar");
?>
```

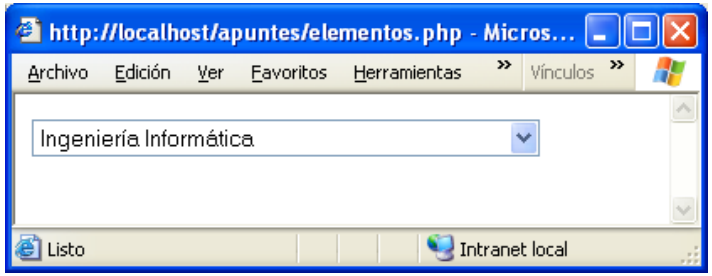


3. ACCESO A ELEMENTOS FORMULARIOS

SELECT simple

```
<SELECT NAME="titulacion">
   <OPTION VALUE="II" SELECTED>Ingeniería Informática
   <OPTION VALUE="ITIG">Ingeniería Técnica en Informática de Gestión
   <OPTION VALUE="ITIS">Ingeniería Técnica en Informática de Sistemas
</SELECT>
```

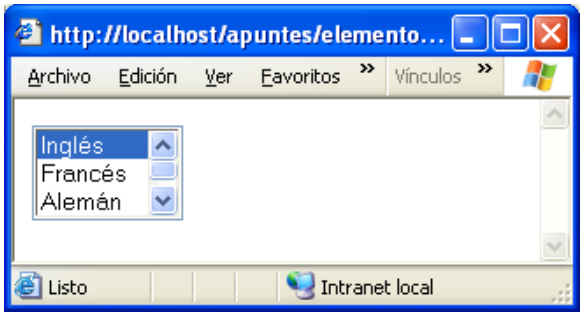
```
<?PHP
   print ($titulacion);
   //print ($_REQUEST ['titulacion']);
?>
```



3. ACCESO A ELEMENTOS FORMULARIOS

SELECT múltiple

```
<SELECT MULTIPLE SIZE="3" NAME="idiomas[]">
   <OPTION VALUE="ingles" SELECTED>Inglés
   <OPTION VALUE="frances">Francés
   <OPTION VALUE="aleman">Alemán
   <OPTION VALUE="holandes">Holandés
</SELECT>
```

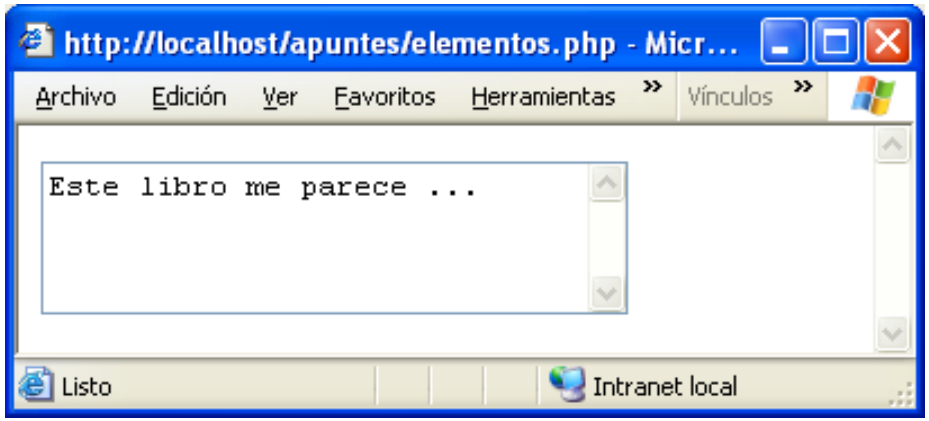



```
<?PHP
   $n = count ($idiomas);
   for ($i=0; $i<$n; $i++)                  //foreach ($_REQUEST['idiomas'] as $idioma)
      print ("$idiomas[$i]<BR>\n");              //print ("$idioma<BR>\n");
?>
```

3. ACCESO A ELEMENTOS FORMULARIOS

TEXTAREA

```
<TEXTAREA COLS=“30" ROWS=“4" NAME="comentario">
        Este libro me parece ...
</TEXTAREA>

<?PHP
        print ($comentario);
        //print ($_REQUEST [‘comentario’]);
?>
```



4. FORMULARIO PHP

- La forma habitual de trabajar con formularios en PHP es utilizar un único programa que procese el formulario o lo muestre según haya sido o no enviado, respectivamente
- Ventajas:
  Disminuye el número de ficheros
  Permite validar los datos del formulario en el propio formulario
- Procedimiento:
  si se ha enviado el formulario:
    Procesar formulario
  si no:
    Mostrar formulario
  fsi

4. FORMULARIO PHP

Para saber si se ha enviado el formulario se acude a la variable correspondiente al botón de envío. Si este botón aparece de la siguiente forma en el formulario HTML:

```
<INPUT TYPE="SUBMIT" NAME="enviar“ VALUE="procesar">
```

entonces la condición anterior se transforma en:

```
if (isset($enviar))
```

o bien

```
if ($enviar == “procesar”)
```

4. FORMULARIO PHP

Institut Joan d'Àustria

```
<?php
    if ($_POST["submit"])
        echo "<h2>You clicked Submit!</h2>";
    else if ($_POST["cancel"])
        echo "<h2>You clicked Cancel!</h2>";
?>
<form action="form.php" method="post">
    <input type="submit" name="submit" value="Submit">
    <input type="submit" name="cancel" value="Cancel">
</form>
```

4. FORMULARIO PHP

Institut Joan d'Àustria

VALIDACIÓN DE FORMULARIOS

- Toda la información proveniente de un formulario debe considerarse por norma como contaminada, y hay que validarla antes de darla por buena y procesarla
- Lo más eficiente es mostrar los errores sobre el propio formulario para facilitar su corrección.
- Procedimiento:

```
si se ha enviado el formulario:
   si hay errores:
      Mostrar formulario con errores
   si no:
      Procesar formulario
   fsi
si no:
   Mostrar formulario
fsi
```

4. FORMULARIO PHP

Institut Joan d'Àustria

VALIDACIÓN DE FORMULARIOS

- Este procedimiento se puede resumir para que sólo haya que mostrar una vez el formulario, bien con los valores por defecto o con los valores introducidos, y con los errores en su caso:

```
si se ha enviado el formulario:
   validar datos
fsi
si se ha enviado el formulario y no hay errores:
   Procesar formulario
si no:
   Mostrar formulario con valores por defecto o ya enviados
fsi
```

5. SUBIDA DE FICHEROS AL SERVIDOR

Para subir un fichero al servidor se utiliza el elemento de entrada FILE

Hay que tener en cuenta una serie de consideraciones importantes:

- El elemento FORM debe tener el atributo ENCTYPE="multipart/form-data“
- El fichero tiene un límite en cuanto a su tamaño. Este límite se fija de dos formas diferentes:

  En el fichero de configuración php.ini

  En el propio formulario

5. SUBIDA DE FICHEROS AL SERVIDOR

Institut Joan d'Àustria

php.ini

```
;;;;;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;;;;;
; Whether to allow HTTP file uploads.
file_uploads = On

; Temporary directory for HTTP uploaded files (will use
; system default if not specified).
;upload_tmp_dir =

; Maximum allowed size for uploaded files.
upload_max_filesize = 2M
```

formulario

```
<INPUT TYPE=”HIDDEN” NAME=”MAX_FILE_SIZE” VALUE='102400'>
<INPUT TYPE=”FILE” NAME="fichero">
```

5. SUBIDA DE FICHEROS AL SERVIDOR

Consideraciones (cont)

Debe darse al fichero un nombre que evite coincidencias con ficheros ya subidos. Por ello, y como norma general, debe **descartarse el nombre original** del fichero y crear uno nuevo que sea único

El fichero subido se almacena en un directorio temporal y hemos de moverlo al directorio de destino usando la función move_upload_file()

Procedimiento:

si se ha subido correctamente el fichero:
    Asignar un nombre al fichero
    Mover el fichero a su ubicación definitiva
si no:
    Mostrar un mensaje de error
fsi

5. SUBIDA DE FICHEROS AL SERVIDOR

```
<INPUT TYPE="HIDDEN" NAME="MAX_FILE_SIZE" VALUE="102400">
<INPUT TYPE="FILE" SIZE="44" NAME="imagen">
```
HTML

La variable $_FILES contiene toda la información del fichero subido:

$_FILES['imagen']['name']
Nombre original del fichero en la máquina cliente

$_FILES['imagen']['type']
Tipo mime del fichero. Por ejemplo, "image/gif"

$_FILES['imagen']['size']
Tamaño en bytes del fichero subido

$_FILES['imagen']['tmp_name']
Nombre del fichero temporal en el que se almacena el fichero subido en el servidor

$_FILES['imagen']['error']
Código de error asociado al fichero subido

5. SUBIDA DE FICHEROS AL SERVIDOR

```
if (is_uploaded_file ($_FILES['imagen']['tmp_name']))
{
   $nombreDirectorio = "img/";
   $idUnico = time();
   $nombreFichero = $idUnico ."-".$_FILES['imagen']['name'];
   move_uploaded_file ($_FILES['imagen']['tmp_name'],
                $nombreDirectorio . $nombreFichero);
}
else
   print ("No se ha podido subir el fichero\n");
```

6. VALIDACION DE FORMULARIOS

Una vez recibidos los datos de un formulario y antes de comenzar a utilizarlos, es necesario comprobar si se corresponden con lo esperado, para procesarlo sin error:

- Si no es vacío,
- Si es del tipo esperado (número entero o decimal, texto, etc.).

Existen una serie de familias de funciones que permiten comprobar la existencia, el tipo o el contenido de un dato:

- funciones is_
- funciones ctype_
- funciones filter_
- funciones _exists
- expresiones regulares

6.1. FUNCIONES IS_

La funciones is_ son un conjunto de funciones booleanas que devuelven **true** si el argumento es de un tipo de datos determinado y **false** si no lo son.

	Funcion	Tipo de datos	funciones equivalentes
Existencia	isset($valor)	Devuelve si el dato esta definido o no	
	is_null($valor)	null	
	is_bool($valor)	booleano	
	is_numeric($valor)	numero	
Numeros	is_int($valor)	entero	is_integer($valor) is_long($valor)
	is_float($valor)	float	is_double($valor)
cadenas	Is_string($valor)	cadena	

6.1. FUNCIONES IS_

	Funcion	Tipo de datos
Otros	is_scalar($valor)	Escalar (entero, float, cadena o booleano)
	is_array($valor)	matriz
	is_callable($valor)	funcion
	is_object($valor)	object
	is_resource($valor)	recurso
	is_countable($valor)	contable (matriz u objeto que implementa Countable)
	is_iterable($valor)	iterable (matriz u objeto que implementa Traversable)

6.1. FUNCIONES IS_

Funciones is_

Estas funciones son en general de poca utilidad con datos provenientes de un formulario. Estas funciones comprueban el tipo de los datos y la información que llega de un formulario es siempre del tipo cadena. Las dos excepciones serían:

- **is_numeric($valor)**: evalúa si el argumento se puede interpretar como número (aunque sea cadena). Esta función es ideal para comprobar si un dato recibido de un formulario es un número.
- **is_int($valor) o is_float($valor)** sólo hacen comprobaciones sobre el tipo de los datos y devolverían siempre false.
- **isset()** evalúa si el argumento está o no definido, independientemente del tipo

6.2. FUNCIONES CTYPE_

Son un conjunto de funciones booleanas que devuelven si todos los caracteres de una cadena son de un tipo determinado

Función	Tipo de datos
ctype_alnum($valor)	alfanuméricos
ctype_alpha($valor)	alfabéticos (mayúsculas o minúsculas, con acentos, ñ, ç, etc)
ctype_cntrl($valor)	caracteres de control (salto de línea, tabulador, etc)
ctype_digit($valor)	dígitos
ctype_graph($valor)	caracteres imprimibles (excepto espacios)
ctype_lower($valor)	minúsculas
ctype_print($valor)	caracteres imprimibles
ctype_punct($valor)	signos de puntuación (caracteres imprimibles que no son alfanuméricos ni espacios en blanco)
ctype_space($valor)	espacios en blanco (espacios, tabuladores, saltos de línea, etc)
ctype_upper($valor)	mayúsculas
ctype_xdigit($valor)	dígitos hexadecimales

6.2. FUNCIONES CTYPE_

Estas funciones se pueden aplicar a los datos recibidos de un formulario ya que hacen comprobaciones de todos los caracteres de una cadena. La más útil es:

- **ctype_digit($valor)**: Evalúa si todos los caracteres del argumento son dígitos (0, 1, …, 9), da false si no lo son. Por lo que permite identificar si el dato recibido es un entero positivo (sin punto decimal ni signo negativo)

```php
<?php

    $numero = $_REQUEST["numero"];

    if (ctype_digit($numero)) {
        print "<p>Ha escrito un entero positivo: $numero.</p>\n";
    } else {
        print "<p>NO ha escrito un entero positivo: $numero.</p>\n";
    }

?>
```

6.3. FUNCIONES FILTER_

La función filter más simple y utilizada es la función **filter_var($valor [, $filtro [, $opciones]])**, que devuelve los datos filtrados o false si el filtro falla.

Filtro	Tipo de datos
FILTER_VALIDATE_INT	entero
FILTER_VALIDATE_BOOLEAN	booleano
FILTER_VALIDATE_FLOAT	float
FILTER_VALIDATE_REGEXP	expresión regular
FILTER_VALIDATE_DOMAIN	dominio web
FILTER_VALIDATE_URL	URL no internacionalizada
FILTER_VALIDATE_EMAIL	dirección de correo
FILTER_VALIDATE_IP	dirección IP
FILTER_VALIDATE_MAC	dirección MAC física

6.3. FUNCIONES FILTER_

Ejemplo: Uso de la función filter_var() con el argumento FILTER_VALIDATE_INT.

```php
<?php
    $numero = $_REQUEST["numero"];

    if (filter_var($numero, FILTER_VALIDATE_INT)) {
        print "<p>Ha escrito un número entero: $numero.</p>\n";
    } else {
        print "<p>NO ha escrito un número entero: $numero.</p>\n";
    }
?>
```

6.4. FUNCIONES XXX_EXISTS()

Institut Joan d'Àustria

Función function_exists()

Devuelve si la función existe o no. Algunas funciones no existen en versiones antiguas de PHP que todavía se utilizan, por lo que puede ser conveniente comprobar si una función existe antes de utilizarla.

```php
<?php
print "<pre>"; print_r($_REQUEST); print "</pre>\n";

$numero = $_REQUEST["numero"];

if (function_exists("filter_var")) {
    if (filter_var($numero, FILTER_VALIDATE_INT)) {
        print "<p>Ha escrito el número entero $numero.</p>\n";
    } else {
        print "<p>NO ha escrito un número entero.</p>\n";
    }
} else {
    print "<p>La función <strong>filter_var</strong> "
        . "no está disponible en este servidor.</p>\n";
}
?>
```

6.4. FUNCIONES XXX_EXISTS()

Función array_key_exists()

La función booleana **array_key_exists($indice, $matriz)** devuelve si un elemento determinado de una matriz existe o no.

Para comprobar si existe un elemento de una matriz también se puede utilizar la función **isset()** comentada en el apartado siguiente

6.5. EXPRESIONES REGULARES

Las expresiones regulares permiten definir patrones de coincidencia y aplicarlas a cadenas de texto para saber si la cadena (o parte de ella) cumple el patrón e incluso realizar transformaciones de la cadena.

- En los años 90, PHP utilizaba las expresiones regulares POSIX extendido, pero a partir de PHP 5.3.0 (junio de 2009) las expresiones regulares POSIX extendido se consideran obsoletas.
- Desde PHP 4.2.0 (abril de 2002), PHP cuenta con las expresiones regulares compatibles con Perl (en inglés, PCRE), que siguen la sintaxis y semánticas del lenguaje de programación Perl 5.
- PHP 4.2.0 y posteriores incluyen la biblioteca de código libre escrita en C PCRE (Perl Compatible Regular Expressions).

6.5. EXPRESIONES REGULARES

Institut Joan d'Àustria

Funciones de expresiones regulares compatibles con Perl

preg_match($patron, $cadena [, $matriz_coincidencias [, $modificadores [, $desplazamiento]]]) compara una cadena con un patrón y devuelve 1 si el patrón ha coincidido o 0 si no.

- La primera coincidencia encontrada se puede guardar en el argumento opcional $matriz_coincidencias y, si se añade el modificador **PREG_OFFSET_CAPTURE**, se guarda también en el argumento opcional $matriz_coincidencias la posición de la coincidencia encontrada.
- El argumento opcional $desplazamiento es un número que permite indicar en qué carácter se inicia la búsqueda.
- Los patrones deben empezar y acabar con el carácter / (barra).

6.5. EXPRESIONES REGULARES

Institut Joan d'Àustria

Ejemplo expresiones regulares en PHP:

```php
<?php
    $cadena1 = "1234567";
    $cadena2 = "abcdefg";
    $patron = "/^[[:digit:]]+$/";

    if (preg_match($patron, $cadena1)) {
        print "<p>La cadena $cadena1 son sólo números.</p>\n";
    } else {
        print "<p>La cadena $cadena1 no son sólo números.</p>\n";
    }

    if (preg_match($patron, $cadena2)) {
        print "<p>La cadena $cadena2 son sólo números.</p>\n";
    } else {
        print "<p>La cadena $cadena2 no son sólo números.</p>\n";
    }
?>
```